



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

Address: COMMISSIONER FOR PATENTS

P.O. Box 1450

Alexandria, Virginia 22313-1450

www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/829,668	04/22/2004	Douglas C. Burger	119832-163869	6831
60172 7590 07/23/2010 SCHWABE, WILLIAMSON & WYATT, P.C. 1420 FIFTH, SUITE 3400 SEATTLE, WA 98101-4010				
EXAMINER				
FENNEMA, ROBERT E				
ART UNIT		PAPER NUMBER		
2183				
MAIL DATE		DELIVERY MODE		
07/23/2010		PAPER		

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

# Office Action Summary

## Application No.

10/829,668

## Applicant(s)

BURGER ET AL.

## Examiner

Robert Fennema

## Art Unit

2183

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --  
Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

## Status

- 1) ☒ Responsive to communication(s) filed on 03 June 2010.
- 2a) ☒ This action is **FINAL**. 2b) ☐ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

## Disposition of Claims

- 4) ☒ Claim(s) 37-44, 46-50, 52-58 and 60 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 37-44, 46-50, 52-58, and 6 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

## Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on \_\_\_\_\_ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

## Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
  - ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
  - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

## Attachment(s)

- 1) ☐ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
- 3) ☐ Information Disclosure Statement(s) (PTO/SB06)  
Paper No(s)/Mail Date \_\_\_\_\_
- 4) ☐ Interview Summary (PTO-413)  
Paper No(s)/Mail Date \_\_\_\_\_
- 5) ☐ ~~Notice of Informal Patent Application~~
- 6) ☐ Other: \_\_\_\_\_

**DETAILED ACTION**

1. Claims 37-44, 46-50, 52-58, and 60 are pending. Claims 37-38, 43-44, 46-50, 52-55, 57-58, and 60 amended as per Applicant's request.
2. Examiner notes the entry of the following papers:
  - Amendment to the claims and remarks filed 6/3/2010

***Claim Objections***

3. In Claim 37, Line 10, Claim 47, Line 13, Claim 57, Line 15, and Claim 60, Line 29, "prior to the respective associated operands of the subset of the group of instructions are available" does not make grammatical sense, and must be corrected. Examiner has interpreted the claims as "prior to availability of the respective associated operands of the subset of the group of instructions", however, correction or clarification in the next response is required.
4. In Claim 38, Line 4, there are two commas where there should be only one.
5. In Claim 46, Line 4, it is unclear if the "s" at the end of "frames" was intended to be deleted or not. Applicant is reminded of the MPEP which indicates that amendments of under three characters, or amendments which are difficult to perceive, should be indicated by single brackets, and not strikethrough. Examiner is interpreting the "s" as being deleted, but clarification is required.

6. In Claims 50, and 52-55, Applicant recites a "computer-readable medium", which lacks antecedent basis, as the previous claims recite a "computer-accessible medium".
7. In Claim 57, Line 12, "preselect" should read "preselected".

***Claim Rejections - 35 USC § 101***

8. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 47-50 and 52-55 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. The claims recite a computer-accessible medium, which has not been specifically defined in the specification, only as a combination of the definitions provided on Page 14 of the specification, which includes both statutory storage mediums, and non-statutory communications mediums. To overcome this rejection, Applicant can either amend the claims to recite a computer-readable storage medium, instead of a computer-accessible medium, or to recite a "non-transitory computer-accessible medium", which excludes all non-statutory embodiments such as carrier waves or signals, or cancel the claims.

***Claim Rejections - 35 USC § 103***

9. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

10. Claims 37-39, 43-44, 46-48, 50, 52-58, and 60 are rejected under 35

U.S.C. 103(a) as being unpatentable over Requa et al. ("The Piecewise Data Flow Architecture: Architectural Concepts", herein Requa), in view of Patterson et al. ("Computer Architecture, A Quantitative Approach", herein Patterson).

11. As per Claim 37, Requa teaches: A method, comprising:

assigning a group of instructions selected from the plurality of groups of instructions partitioned from a program to a subset of interconnected computation nodes preselected from a plurality of interconnected computation nodes (Page 426, first column, second paragraph, the blocks are sent to processors), the instructions having respective associated operands (Page 427, for instructions to have dependencies, they must have operands), and each computation node includes an execution unit having one or more arithmetic units, floating point units, memory address units, or branch units (Figure 1, Page 427. In order for a processor to execute an instruction, it inherently requires execution units, and each of the scalar processor, memory processor, and SIMD processor would necessarily have at least one of the claimed units in order to have any functionality); and

executing the subset of instructions as each one of the instructions in the subset of instructions loaded into the frame of buffers receives the respective associated operands for execution (Page 433, first column, third paragraph, an instruction waits for

input operands, then is executed. Also see Page 435, Instruction Issue section. When the instruction in the list has all required operands, it is sent to the instruction-issue section, and then a processor), but fails to teach:

each computation node includes a store;

loading a subset of instructions of the assigned group of instructions into a frame of buffers comprising the stores disposed on the preselected subset of interconnected computation nodes having been assigned the group of instructions, wherein the loading is performed prior to the associated operands of the subset of instructions are available; and

receiving the respective associated operands of the subset of instructions by the preselected subset of computation nodes, including a first computation node of the preselected subset of computation nodes directly receiving a first associated operand of a first instruction loaded into the first computation node, from a second computation node of the preselected subset of computation nodes, wherein the first computation node has an input port capable of being coupled to the second computation node to enable the directly receiving of the first associated operand.

Requa teaches loading instructions into a centralized instruction buffer, and only when all operands are available for execution, is it sent to the appropriate processor for execution, thus, Requa does not teach a buffer for each node, receiving instructions prior to the operands for said instructions being present. However, Patterson teaches of two methods to allow for dynamic scheduling, which has the advantage of removing

multiple hazards: scoreboards and reservation stations. Requa's system is essentially a scoreboard, a centralized buffer that keeps track of when instruction operands are available in the register file, then issues appropriate. However, Patterson also suggests a reservation station, which has additional advantages in that it removes additional WAW and WAR hazards (Page 252), by using register renaming, and furthermore, distributes the logic by having a reservation station in each functional unit, instead of in one centralized location (Page 252), and further does not require data to be written into the register file before it can be accessed by the instruction. The advantages of the distribution are explained on Page 257 of Patterson, where by having each functional unit (processor) having its own reservation station, when results are broadcast, multiple instructions can be issued at once, wherein a centralized issue buffer cannot do so, in addition to the other advantages listed above. Additionally, a reservation station system such as Patterson's allows each processor to directly send data to another processor, via the bus, without the data first being stored in the register file. Therefore, given the aforementioned advantages, one of ordinary skill in the art would have been motivated to implement reservation station logic in Requa's architecture, which would involve both distribution of instruction buffers into each functional unit, and the issuing of instructions to those buffers prior to the operands being available.

12. As per Claim 38, Requa teaches: The method of claim 37, further comprising storing the first associated operand in a first store of the first computation node, wherein

the first store is coupled to the input port (PDF, paragraph 1, input operands are stored in registers),

storing the first instruction in a second store of the first computation nodes, wherein the second store is coupled to an instruction sequencer (Page 427, one of the processor-specific FIFOs, or in Patterson, the reservation station for each node also holds instructions),

matching the first associated operand with the first instruction by an instruction wakeup unit (PDF, paragraph 1, the operand source fields are modified as data comes in),

executing the first instruction by an execution unit of the first computation nodes using at least the first associated operand to produce output data (PDF, paragraph 1, the instructions are executed after receiving inputs),

routing the output data to an output port of the first computation nodes, wherein the output port is capable of being coupled to a third of the preselected subset of interconnected computation nodes (Figure 1 and Page 427, second column, paragraph 2. Any consumer can receive any data from the interconnection network, thus all processors are capable of sending data to any other processor) to directly provide the output data to the third computation node (Patterson, Page 254, any processor can receive results directly from any other processor via the bus).

13. As per Claim 39, Requa teaches: The method of claim 37, wherein at least one of the plurality of groups of instructions is a basic block (Page 426, first column third

paragraph).

14. As per Claim 43, Patterson teaches: The method of claim 37, wherein loading the group of instructions into a frame of buffers comprising the stores disposed on the preselected subset of interconnected computation nodes includes:

    sending at least two instructions selected from the group of instructions from an instruction sequencer to a selected one of the preselected subset of interconnected computation nodes for storage in a store of the selected computation node, wherein the sending is performed prior to the at least two instructions having all necessary associated operands for execution (Page 252).

15. As per Claim 44, Requa teaches: The method of claim 37, wherein executing the subset of instructions loaded into the frame of buffers as each one of the instructions in the subset of instructions receives the respective associated operands for execution includes:

    matching at least one instruction selected from the group of instructions with at least one operand (Page 428, detecting data dependencies).

16. As per Claim 46, Requa teaches: The method of claim 37, further comprising concurrently assigning another group of instructions selected from the plurality of groups of instructions to another preselected subset of interconnected computation nodes for concurrent execution using another frame of buffers comprising stores

disposed on the another preselected subset of interconnected computation nodes (Page 436, Second Column, Second and Third paragraphs. Requa discusses that blocks can overlap each other as long as they do not need results from each other. Additionally, in the third paragraph, Requa discusses that the PDF architecture is capable of supporting multiple program executions simultaneously, which would also read on the limitation):

wherein the two groups of instructions are capable of concurrent execution (Page 436, Second Column, Second and Third paragraphs. Requa discusses that blocks can overlap each other as long as they do not need results from each other. Additionally, in the third paragraph, Requa discusses that the PDF architecture is capable of supporting multiple program executions simultaneously, which would also read on the limitation).

17. As per Claim 47, Requa teaches: An article comprising a computer-accessible medium having computer executable codes stored therein, configured to enable a machine, in response to execution of the codes by the machine, to:

assign a group of instructions selected from a plurality of groups of instructions partitioned from a program, to a subset of interconnected computation nodes preselected from a plurality of interconnected computation nodes of the machine (Page 426, first column, second paragraph, the blocks are sent to processors), the instructions having respective associated operands (Page 427, for instructions to have dependencies, they must have operands), and each computation node includes an execution unit having one or more arithmetic logic units, floating point units, memory

address units, or branch units (Figure 1, Page 427. In order for a processor to execute an instruction, it inherently requires execution units, and each of the scalar processor, memory processor, and SIMD processor would necessarily have at least one of the claimed units in order to have any functionality); and

load a subset of the group of instructions into a frame of buffers (Page 426, first column, second paragraph, also see Page 433, "The PDF Block Processor". Specifically, see Page 434, Figure 9. The Instruction List, as it is used by all processors, and is a buffer, it is a buffer which spans all the connected nodes/processor),

wherein the loaded instructions are executed as each one of the instructions receives the respective associated operands for execution (Page 433, first column, third paragraph, an instruction waits for input operands, then is executed. Also see Page 435, Instruction Issue section. When the instruction in the list has all required operands, it is sent to the instruction-issue section, and then a processor), but fails to teach:

each computation node includes a store;

a frame of buffers comprising the stores disposed on the preselected subset of interconnected computation nodes,

wherein the subset of the group of instructions is loaded into the frame of buffers prior to the respective associated operands of the instructions are available; and

wherein the receiving of respective associated operands includes at least a first computation node of the preselected subset of interconnected computation nodes directly receiving a first associated operand of a first instruction from a second computation node of the preselected subset of interconnected computation nodes.

Requa teaches loading instructions into a centralized instruction buffer, and only when all operands are available for execution, is it sent to the appropriate processor for execution, thus, Requa does not teach a buffer for each node, receiving instructions prior to the operands for said instructions being present. However, Patterson teaches of two methods to allow for dynamic scheduling, which has the advantage of removing multiple hazards: scoreboards and reservation stations. Requa's system is essentially a scoreboard, a centralized buffer that keeps track of when instruction operands are available in the register file, then issues when appropriate. However, Patterson also suggests a reservation station, which has additional advantages in that it removes additional WAW and WAR hazards (Page 252), by using register renaming, and furthermore, distributes the logic by having a reservation station in each functional unit, instead of in one centralized location (Page 252), and further does not require data to be written into the register file before it can be accessed by the instruction. The advantages of the distribution are explained on Page 257 of Patterson, where by having each functional unit (processor) having its own reservation station, when results are broadcast, multiple instructions can be issued at once, wherein a centralized issue buffer cannot do so, in addition to the other advantages listed above. Additionally, a reservation station system such as Patterson's allows each processor to directly send data to another processor, via the bus, without the data first being stored in the register file. Therefore, given the aforementioned advantages, one of ordinary skill in the art would have been motivated to implement reservation station logic in Requa's architecture, which would involve both distribution of instruction buffers into each

functional unit, and the issuing of instructions to those buffers prior to the operands being available.

18. As per Claim 48, Requa teaches: The article of claim 47, wherein the computer executable codes, further enable the machine, in response to execution of the codes by the machine, to partition the program into the plurality of groups of instructions during compilation of the program (Page 429, first column, "PDF Architecture").

19. As per Claim 50, Requa teaches: The article of claim 47, wherein the computer-readable medium further includes codes, configured to enable the machine, in response to execution of the codes by the machine, to:

statically assign another group of instructions selected from the plurality of groups of instructions to another preselected subset of interconnected computation nodes of the plurality of interconnected computation nodes for execution (Page 432, see Figure 8, and column 1, paragraph 2).

20. As per Claim 52, Requa teaches: The article of claim 47, wherein the computer-readable medium further includes codes, configured to enable the machine, in response to execution of the codes by the machine, to:

generate a wakeup token to reserve an output data channel of the second computation node to directly route the first associated operand from the second computation node to the first computation node (PDF, Page 427, section column,

second paragraph, also see Patterson, Page 254).

21. As per Claim 53, Requa teaches: The article of claim 47, wherein computer-readable medium further includes codes, configured to enable the machine, in response to execution of the codes by the machine, to:

to repeat said loading until the entire group of instructions are executed (Page 433, in order to execute a block, this clearly has to occur),

to detect execution termination of the group of instructions (Page 433, second column first paragraph, a flag is set when an execution is done, also see Page 430, second paragraph); and

to commit architecturally visible data resulting from execution of the group of instructions to a register file (Page 430, second paragraph).

22. As per Claim 54, Requa teaches: The article of claim 47, wherein the computer-readable medium further includes codes, configured to enable the machine, in response to execution of the codes by the machine, to repeat said loading until the entire group of instructions are executed (Page 433, in order to execute a block, this clearly has to occur),

to detect execution termination of the group of instructions (Page 433, second column first paragraph, a flag is set when an execution is done, also see Page 430, second paragraph); and

to commit architecturally visible data resulting from execution of the group of instructions to a memory (Page 430, second paragraph).

23. As per Claim 55, Requa teaches: The article of claim 47, wherein the computer-readable medium further includes codes, configured to enable the machine, in response to execution of the codes by the machine, to route an output datum arising from executing one of the subset of instructions to a consumer node included in the preselected subset of interconnected computation nodes, wherein an address of the consumer node is included in a token associated with at least one instruction included in the subset of instructions (Page 429, second column, second paragraph).

24. As per Claim 56, Requa teaches: The method of claim 37 further comprising repeating said loading and executing until the entire group of instructions have been executed (Page 433, in order to execute a block, this clearly has to occur).

25. As per Claim 57, Requa teaches: An apparatus, comprising:  
a plurality of interconnected computation nodes (Page 427, Figure 1, the Scalar, Memory, and SIMD Processors), each computation node including an execution unit having one or more arithmetic logic units, floating point units, memory address units, or branch units (Figure 1, Page 427. In order for a processor to execute an instruction, it inherently requires execution units, and each of the scalar processor, memory

processor, and SIMD processor would necessarily have at least one of the claimed units in order to have any functionality); and

a storage medium coupled to a processor and configured to have first instructions stored therein to be executed by the processor (Page 427, Main memory), wherein the first instructions are configured to enable the apparatus, in response to execution of the first instructions, to:

assign a group of second instructions selected from a plurality of groups of second instructions partitioned from a program to a preselected subset of the plurality of interconnected computation nodes, the second instructions having respective associated operands (Page 426, first column, second paragraph, the blocks are sent to processors); and

wherein the subset of second instructions are executed as each one of the subset of second instructions loaded into the frame of buffers receives the respective associated operands for execution (Page 433, first column, third paragraph, an instruction waits for input operands, then is executed. Also see Page 435, Instruction Issue section. When the instruction in the list has all required operands, it is sent to the instruction-issue section, and then a processor), but fails to teach:

each computation node including a store;

wherein at least a first of the associated operands of a first of the second instructions loaded into a first computation node of the preselected subset of the interconnected computation nodes is directly received from a second computation node of the preselected subset of the interconnected computation nodes,

causing a subset of the assigned group of second instructions to be loaded into a frame of buffers comprising the stores disposed on the preselected subset of interconnected computation nodes having been assigned the group of second instructions, wherein the loading is caused prior to the respective associated operands of the subset of the second instruction are available.

Requa teaches loading instructions into a centralized instruction buffer, and only when all operands are available for execution, is it sent to the appropriate processor for execution, thus, Requa does not teach a buffer for each node, receiving instructions prior to the operands for said instructions being present. However, Patterson teaches of two methods to allow for dynamic scheduling, which has the advantage of removing multiple hazards: scoreboards and reservation stations. Requa's system is essentially a scoreboard, a centralized buffer that keeps track of when instruction operands are available in the register file, then issues when appropriate. However, Patterson also suggests a reservation station, which has additional advantages in that it removes additional WAW and WAR hazards (Page 252), by using register renaming, and furthermore, distributes the logic by having a reservation station in each functional unit, instead of in one centralized location (Page 252), and further does not require data to be written into the register file before it can be accessed by the instruction. The advantages of the distribution are explained on Page 257 of Patterson, where by having each functional unit (processor) having its own reservation station, when results are broadcast, multiple instructions can be issued at once, wherein a centralized issue buffer cannot do so, in addition to the other advantages listed above. Additionally, a

reservation station system such as Patterson's allows each processor to directly send data to another processor, via the bus, without the data first being stored in the register file. Therefore, given the aforementioned advantages, one of ordinary skill in the art would have been motivated to implement reservation station logic in Requa's architecture, which would involve both distribution of instruction buffers into each functional unit, and the issuing of instructions to those buffers prior to the operands being available.

26. As per Claim 58, Requa teaches: The apparatus of claim 57, wherein at least one of the plurality of groups of the second instructions is a selected of one a basic block, a hyperblock or a superblock (Page 426, first column third paragraph).

27. As per Claim 60, Requa teaches: A system, comprising:

a plurality of interconnected computing nodes configured to be pre-selectable to cooperatively execute a group of instructions (Page 427, Figure 1, the Scalar, Memory, or SIMD Processors), wherein the group of instructions is one of a plurality groups of instructions partitioned from a program (Page 426, blocks) and the instructions have respective associated operands (Page 427, for instructions to have dependencies, they must have operands);

wherein a first computation node of the plurality of interconnected computing nodes includes:

a computing resource including an execution unit configured to execute instructions (Page 427, Figure 1, the Scalar, Memory, or SIMD Processor), the execution unit having one or more arithmetic logic units, floating point units, memory address units, or branch units (Figure 1, Page 427. In order for a processor to execute an instruction, it inherently requires execution units, and each of the scalar processor, memory processor, and SIMD processor would necessarily have at least one of the claimed units in order to have any functionality); and

an interconnect resource coupled to the computing resource to enable the first computing node to be preselected with at least a second of the plurality of interconnected computation nodes to execute the group of instructions by successively executing subgroups of the group of instructions (Page 427, Second Column, Second paragraph, all processors/nodes are connected to each other through an interconnection network. Page 433 discloses successive execution of parts of a block);

wherein the interconnect resource includes:

an input port capable of coupling the computing resource to the second computation node, and the input port is configured to receive input data (Page 427, Second Column, Second paragraph, every node is connected to every other node, requiring an input port),

a first store coupled to the input port, and configured to store the input data (Page 427, Second Column, Second paragraph, the FIFO queue),

a second store coupled to the execution unit, and configured to receive and store an instruction of a subset of the group of instructions, the second store being a part of a

frame of buffers spanning the preselected subset of interconnected computation nodes to store the subset of the group of instructions loaded into the frame of buffers, wherein the subset of instructions are loaded prior to the respective associated operands of the subgroup of instructions are available (Page 426, first column, second paragraph, also see Page 433, "The PDF Block Processor". Specifically, see Page 434, Figure 9. the Instruction List, as it is used by all processors, and is a buffer, it is a buffer which spans all the connected nodes/processor, and the instructions do not necessarily have their operands when first entered into the buffer),

an instruction wakeup unit to match the input data to the stored instruction (PDF, paragraph 1, the operand source fields are modified as data comes in),

an output port coupled to the execution unit and capable of coupling the computing resource to the second or a third of the preselected subset of interconnected computation nodes (Page 427, Second Column, Second paragraph, every node is connected to every other node, requiring an output port), and

a router coupled to the execution unit, and configured to direct an output data of the execution unit to the output port for provision to one of the second or third computation node (Page 427, the FIFO queue also acts as a kind of router), but fails to teach:

the input port, output port, and router directly coupling the resource to a node for receiving or sending data.

Requa does not teach that each node is directly connected to each other, in order to transfer data from one to the other, they must all transfer data into the

instruction result registers (the register file) before being able to send data to another processor. However, as is well known in the art, Patterson describes on Page 146 that this is what is known as a data hazard, and causes stalls or incorrect performance to occur. Patterson discloses that an easy and well known solution to this problem is data forwarding (Page 147-148), where instead of having to send data to the register file and into the functional units, one can just create a direct line path between the output of a unit and the input, then saving several clock cycles of waiting for the result. Given that data forwarding is well known in the art to avoid the problem of data hazards, one of ordinary skill in the art would have been motivated to allow for direct connections between the different nodes of Requa in order to overcome the deficiency in the architecture.

28. Claims 40-41 are rejected under 35 U.S.C. 103(a) as being unpatentable over Requa and Patterson, in view of Official Notice.

29. As per Claim 40, Requa teaches: The method of claim 37, but fails to teach:  
wherein at least one of the plurality of groups of instructions is a hyperblock.

Requa teaches of a system which uses basic blocks, but does not teach that the groups may be hyperblocks. However, Examiner is taking Official Notice that one of ordinary skill in the art would be capable and motivated to use hyperblocks in lieu of basic blocks, to take advantage of the ability to have multiple exits from a block (While Applicant has not provided a definition of a hyperblock, Examiner has found it to represent a block with one entrance and potentially (but not necessarily) more than one

exit).

30. As per Claim 41, Requa teaches: The method of claim 37, but fails to teach: wherein at least one of the plurality of groups of instructions is a superblock.

Requa teaches of a system which uses basic blocks, but does not teach that the groups may be superblocks. However, Examiner is taking Official Notice that one of ordinary skill in the art would be capable and motivated to use superblocks in lieu of basic blocks, to take advantage of the ability to have multiple exits from a block (While Applicant has not provided a definition of a superblock, Examiner has found it to represent a block with one entrance and potentially (but not necessarily) more than one exit, however, Examiner is unclear how a superblock is different from a hyperblock).

31. Claims 42 and 49 are rejected under 35 U.S.C. 103(a) as being unpatentable over Requa and Patterson, in view of Fisher.

32. As per Claim 42, Requa teaches: The method of claim 37, but fails to teach: wherein at least one of the plurality of groups of instructions is an instruction trace constructed by a hardware trace construction unit at run time.

While Requa teaches the method as disclosed in Claim 37, Requa does not teach about traces, or a trace construction unit to construct such a trace. However, Fisher teaches of Trace Scheduling, where the basic blocks used by Requa are compacted, and instead use traces (Page 462, Section D). The advantage to this

compaction method using traces allows for more efficient parallel code, done in a manner far more efficient than previous methods (Abstract). Given this advantage, one of ordinary skill in the art would have been motivated to use these traces in place of the basic blocks as taught by Requa to further increase the efficiency of the system.

33. As per Claim 49, Requa teaches: The article of claim 47, but fails to teach:

wherein the computer executable codes, further enable the machine, in response to execution of the codes by the machine, to partition the program into the plurality of groups of instructions during run-time.

While Requa teaches the article as disclosed in Claim 47, Requa does not teach that the partitioning of the program is done by a trace mapper. However, Fisher teaches of Trace Scheduling, where the basic blocks used by Requa are compacted, and instead use traces (Page 462, Section D), created and optimized by a scheduler (Page 482, second column, second paragraph). The advantage to this compaction method using traces allows for more efficient parallel code, done in a manner far more efficient than previous methods (Abstract). Given this advantage, one of ordinary skill in the art would have been motivated to use these traces in place of the basic blocks as taught by Requa to further increase the efficiency of the system.

### ***Response to Arguments***

34. Regarding Applicant's amendments to the claims to overcome the rejections, Examiner notes that the issues with the "prior to the operands are available" language in the claims still exist, as the claims do not make grammatical sense as currently written, and while one can attempt to infer what the Applicant is attempting to claim, it should not be left to the imagination, thus the claim should be amended so that it is clear what is being claimed and one does not have to guess as to what the claim is attempting to say. Examiner requests that Applicant either correct the claims in the next response, or explain why the Applicant believes the claims as written are not an issue.

35. Regarding Applicant's amendments to the claims to overcome the 101 rejection, Applicant has not overcome the rejection, as Applicant has simply replaced one undefined term with another undefined term, instead of the Examiners suggested language. Applicant is required to make either of the amendments that the Examiner has suggested, or to cancel the claims. The claims must specifically recite a computer storage medium (or computer-readable storage medium, the key term being that "storage" must be claimed, as that is the only statutory embodiment in the specification), or must clearly indicate that it is a non-transitory medium, any other language will result in a 101 rejection for including non-statutory subject matter as it would encompass the communications media disclosed in the specification.

36. Regarding Applicant's arguments in regard to the art rejections of the claims under 103, Applicant has argued that in Requa, there is no concept at dispatch time that the block of instructions are assigned to be executed by a processor. However, Examiner does not find this argument persuasive, because first, the claims are completely silent towards anything happening at dispatch time, only that the instructions are assigned to a processor, and if the instructions make it to a processor, then clearly, they were assigned there. However, even if such a limitation was claimed, the fact that each processor has a fairly specific purpose would almost require that the dispatch logic does in fact assign instructions to a processor, as a vector instruction is not going to be executing on a scalar processor, and memory instructions are not going to be executing on the SIMD processor, they'll be executed on the memory processor.

Applicant has further argued that Requa's processors are not computation nodes because they allegedly do not contain execution units, and again, Examiner disagrees. The entire purpose of a processor, both in the art and in Requa, is to execute instructions, and Examiner does not believe it is a reasonable argument to say that the processors of Requa do not have execution units to execute instructions, as that would make them completely nonfunctional and unable to perform their disclosed functions. Each processor of Requa is required to have at least one of the claimed functional units, otherwise, they couldn't do any work, and Requa's processor would never do anything, which is clearly not a reasonable interpretation to take of the reference.

Applicant has further argued that Requa does not teach loading a subset of instructions of the assigned group of instructions into a node, however, as Examiner has

stated before, if instructions are sent to a processor, then they clearly must have been assigned there, and since each processor cannot hold simultaneously ever single instruction in the program of potentially thousands or hundreds of thousands of instructions, only a subset of them are loaded at any given time (additionally, the blocks are clearly disclosed as a subset of the entire set of instructions, thus loading a block into a processor is clearly loading a subset of instructions into a node, thus Examiner is unclear what Applicant's issue with the rejection is for this argument).

### ***Conclusion***

37. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Robert Fennema whose telephone number is (571)272-2748. The examiner can normally be reached on Monday-Thursday, 9:30-6:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Eddie P Chan/  
Supervisory Patent Examiner, Art Unit 2183

Robert Fennema  
Examiner  
Art Unit 2183

RF